Amendments to the specification:

Please amend the paragraph beginning on page 10, line 12 to reads as follows.

--A program storage medium readable by a processor, tangibly embodying a program of instructions executed by the processor to perform method steps for ~~processioning~~ processing first header information of a reception packet to provide instruction regarding processing of second header information, wherein the method steps include separating error correction data, payload data and header information from the reception packet; processing in one clock cycle the first header information in a protocol processor unit to provide by the end of the one clock cycle selected instructions for decoding second header information. The method steps may include a payload flag generating step for generating payload flags based on the selected instructions, wherein the payload flags are used by execution units to process the payload data. The selected instructions may be stored in at least one look up table and provided based on results of the processing of the first header information. The second header information may be header information immediately following the first header information.--

Please amend the paragraph beginning on page 16 at line 16 to read as follows.

--A method of processing first header information of a reception packet according to an embodiment of the present application is described with relation to Figure 7. In step **70**, the reception packet is received by a protocol processor. Error

correction data, payload data and header information are separated from the reception packet at step **72**. In step **74**, first header information is processed in one clock cycle. The selected instructions are provided by the end of the one clock cycle in step **76**. The first header information may be a first word of header information of a predetermined bit length. The second header information may be a second word of header information of the predetermined bit length immediately following the first header information. The method described in Figure 7 would be suitable for application in the protocol processor **1** described with reference to Figure 1, for example. While the steps of Figure 7 are shown as separated, the processing occurs substantially simultaneously with reception of the header information. One word of the header information is received every clock cycle and an instruction is executed every clock cycle. It should be noted that proper error correction requires error correction data as well the rest of the reception packet which is checked for errors. After the entire header has been received and processed, payload flags may be output from the protocol processor to guide payload ~~preecessing~~ processing.--

Please amend the paragraph beginning on page 26 at line 20 to read as follows.

--As mentioned above, problems arise when long fields have to be compared by the processor (e.g. destination addresses in IPv6 of 128 bits). The comparison can be split up in time without losing any performance according to an embodiment of the present invention since data will arrive at fixed rate in words of finite length through the dynamic buffer **12** anyway, 32 bits for example. A compare unit **24** in which fields can be split up for comparison is described with reference to Figure 6. By using several vectors

in the PCB **54** for different parts of the field, the comparison can be performed partially and accumulated in the compare unit **24**. The value from the PCB **54** is compared to the extracted value from the packet header which might be only a part of the total data in the header. An instruction from the instruction decoder indicates whether the comparison is a partial comparison. The result of this partial comparison can be used directly to provide instructions for processing the packet in the next clock cycle. The partial comparison result is used if a previous partial comparison result is a "1". If the result of the previous partial comparison result is a "0", then a "0" is output from the comparator. The final result will only be a "1" if all comparisons result in a "1". More specifically, ~~multiplexer~~ multiplexor **60** is controlled using a control signal generated by the combinational logic unit **62** to output the partial comparison result, or a "0". Figure 6B illustrates a truth table corresponding to an example of boolean logic utilized by the combinational logic unit **62**.--